HTML5

A BASIC GUIDE

INTRODUCTION

HTML5 is an emerging standard for distributing content on the Web. While it is still under development, currently a W3C Working Draft, many components of HTML5 are now stable and can be implemented and understood by modern browsers.

DOCUMENT STRUCTURE

Every HTML5 document consists of five essential parts. In order they are the doctype declararion, html, head, title and body elements. In addition, a meta element is also used in the head.

Document Type

The <doctype> declaration for an HTML document has changed to a simplified format from what it was in previous versions of html and xhtml. The new doctype is:

<!DOCTYPE html>

Besides being easy to remember and type, this change allows HTML5 to be backward compatible with previous versions of html.

Root Element

The <html> element is the parent or root element for the entire document. It is the first element to open and the last to close. It is declared or opened immediately after the doctype declaration (be sure there is not an empty line between them). Its closing tag is the last tag on the page. It would look like this:

<html> ... all other page code ... </html>

Head Element

The root element (<html>) has two children: <head> and <body>. The <head> element is opened immediately after the root or <html> element. The content inside the head element is intended to provide information about the page to the browser. It can have many different types of elements inside of it to accomplish this purpose. It appears as:

<head> ... all head element code ... </head>

Meta Element

The <meta> element is a generic element used for describing the document and its content. HTML5 uses a meta element, in the head, to describe for the browser what character encoding is to be used in the page. In common web terminology it is said to be a child element of the head. For HTML5 the meta element markup is:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Page Name | Site Name - Description</title>
    … other meta data here …
  </head>
  <body>
    ... body content here ...
  </body>
</html>
```

BASIC HTML5 DOCUMENT

<meta charset="utf-8">

Title Element

The <title> element provides a title value to the browser. This title is displayed in the browser window's title bar and the tab in browsers that use a tabbed interface. It is very important to provide a value in the title element to assist and clarify the topic of the page to the client (the person viewing your web page). Common title markup looks something like this:

<title>Page Name | Site Name - short description</title>

Body Element

The <body> element encloses all the content that is to be displayed in the browser for the client to use. It should open immediately after the head has closed and should close immediately before the html element's closing tag. The body markup is:

<body>... Content Goes Here ... </body>

ELEMENTS

If you have been around long you have heard html markup referred to as "tags". While it is partially correct, it is also partially wrong. In reality, the markup language of html (in any of its versions) is comprised of elements. Elements themselves are sometimes comprised of opening and closing tags.

For example, if you look at the list of elements that are required for an HTML5 page you will find that all of them, except the meta element, have an opening and closing tag. The opening tag indicates where the element begins. The closing tag indicates where the element ends. It is easily identified by its forward slash preceding the name (e.g. </body>).

However, you will note that the <meta> element has no such closing tag. In this case the meta element is self-closing. All of its information exists within the opening element tag and has no need of a closing tag.

ELEMENT TYPES

HTML5 elements can be categorized in a variety of ways. According to John Allsopp, there appears to be three classifications of elements: structural, content and rhetorical. The W3C breaks elements into nine categories - root, metadata and scripting, embedding, text, grouping, forms, sections, tabular and interactive. However, for beginning coders, I prefer three general classifications: page, structural and content.

All elements, regardless of classification in HTML5 are meant to be ***semantic***. This means that the name of the element carries meaning and suggests its use. This meaning is to enable human readability as well as software functionality.

Page Elements

These elements make the page work in the browser. They do not describe the content or the organization of the content. The essential page elements are: <html>, <head>, <title>, <base>, <link>, <style>, <meta>, <script>, <noscript> and <body>. Note: Not all of these must be used but are available if needed.

Structural Elements

The structural elements give structure to the content. When taking notes in an outline format, you build a structure consisting of indentations, numbers, letters, bullets or other indicators. The outline is the structure used to organize the content, not the content itself. Likewise, in HTML5 there are many structural elements, some of the more common ones are <address>, <article>, <aside>, , <footer>, <header>, <main>, <nav>, <section> and a number of headings - <h1>, <h2>, <h3>, <h4>, <h5>, <h6>.

Content Elements

These elements are, as the name implies, meant to describe the content that they hold. The majority of existing elements are content elements, but less than 30 make up the most frequently used, they are: <a>, <blockquote>, <cite>, <figure>, <figcaption>, <img>, <p>, ; Lists - , <ul>, <dl> and their children - , <dt> and <dd>; Forms - <form> and its children <fieldset>, <legend>, <label>, <input> and <textarea>; Tables - <table> and its children <caption>, <thead>, <tfoot>, <tbody>, <tr>, <th>, <td>, <col> and <colgroup>; and a few meant to change the way text is to be understood: <abbr>, <b>,
, , <i>, <hr> and . For media content HTML5 now offers two new elements <video> and <audio> and their child <source>; and for JavaScript graphics <canvas>.

Each of these elements plays a specific role in adding meaning to the page and the content. For more information on these and other elements you can refer to the table that appears later in this document or to http://html5doctor.com/element-index/ or http://joshduck.com/periodic-table.html. The official list of all HTML5 elements can be seen at: http://dev.w3.org/html5/spec-author-view/index.html#elements-1

EMPTY OR NON-EMPTY

Additionally, elements can be said to be empty (void) or non-empty (non-void). Again, referring to the elements used in our earlier page example, the <html>, <head>, <title> and <body> elements all have an opening and closing tag and some form of content is expected between the opening and closing tags. Therefore, each of these is considered non-empty. On the other hand, the <meta> element, which does not have a closing tag, is said to be empty. All elements fall into one of these two categories.

ELEMENT SYNTAX

Syntax describes the way we write things. In human languages it describes the correct use and order of words to create meaning. When writing HTML5 elements there are also three major syntax rules and a big Best Practice that should be followed:

1. The opening tag and closing tag must match exactly (e.g. <A>... won't work, while <a>... will work).
2. The elements must be spelled correctly (e.g. <style> is great, while <stile> is not).
3. Elements must open and close in proper order (e.g. Stop! He commanded. is wrong, while Stop! He commanded. is right) [The rule of thumb is: first in, last out]
4. Best Practice - make elements lower case, it is easier to remember and avoid mistakes that way.

ATTRIBUTES

An attribute adds descriptive detail to an element; similar to the way we use attributes to describe other people (hair color, height, weight). However, each element allows only certain attributes to be used with it. That being said, there are a large number of attributes that are allowed by many elements. There are even a few attributes that are "global", meaning they can be used with every element. Some of the more common of these global attributes are: class, title, id and style.

View the official list of global attributes:
http://dev.w3.org/html5/spec-author-view/global-attributes.html#global-attributes

View the official list of all attributes:
http://dev.w3.org/html5/spec-author-view/index.html#attributes-1

Attribute Syntax

Just as with elements there are syntax rules that should be followed when writing HTML5 attributes. Within the element's opening tag the name of the attribute should be written, followed by an equal sign, followed by quotes and within the quotes the value of the attribute. For example, if creating a hyperlink to the NASA web site, the code would look like this:

Visit NASA

In this example, the href attribute tells the browser that when the link is clicked this is the destination it should go to. The address of NASA is placed inside the quotes as the address of the destination. The text "Visit NASA" is what the client sees on the screen and clicks on to be redirected to the NASA web site.

However, in HTML5 there are a few (very few) attributes that are called "boolean" attributes. This means that if the attribute exists in the element it is "true" or "on". An example of this is in a form input element, if the "required" attribute is present, it will not allow the form to be submitted if the input doesn't have a value. What it does NOT do is know if the value is correct, just that there is a value present. For example:

<input type="email" name="emailaddress" required>

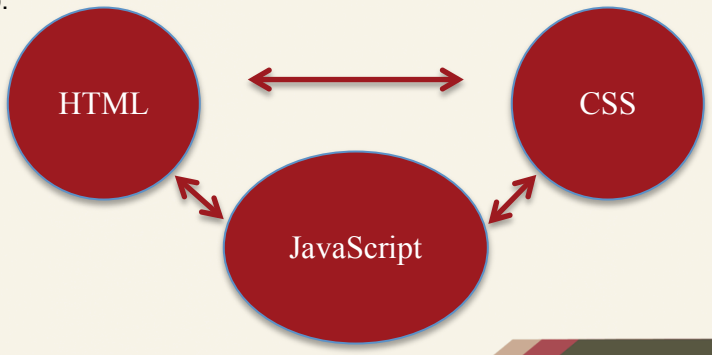
Important Attribute Changes

HTML5 no longer allows attributes to be used to style how the element will appear. For example, if one wanted the text inside of a paragraph to be aligned to the right, in previous versions of html you could use the align attribute and give it a value of "right". HTML5 does NOT allow this. Instead, you should use Cascading Style Sheet (CSS) language to do all presentational markup. This also includes the old font attribute that bloated many a web page.

Web Model

A model is an abstract way of looking at the world. Weather forecasters build models of temperatures, wind patterns, high and low pressures systems in order to tell us what our weather will be tomorrow or next week. We too have a model of how the web should be built so that what we build today will continue to work well in our current browser and in future ones. The model identifies three main components that browsers use to deliver our content - structure, appearance and behavior. The structure is the HTML5 markup code; appearance, as mentioned previously, is the responsibility of CSS; while behavior is most often accomplished using JavaScript.

Our web model says that all three should be built and stored separately and then "linked" to allow the others to share resources or interact to deliver the content in the desired way. Following this model has other benefits: 1) provides organization to your materials, 2) makes things easier to maintain, 3) saves time for the software and the humans responsible for the web site, 4) reduces the amount of code needed and 4) saves money. Any one of these would be reason enough to follow the model, but when all four are understood, it leaves us with no reason not to.



ORDERED LIST

Ordered lists are used when items in the list are prioritized or sequential. The basic building blocks are the element and the element(s) which represent the individual items in the list. For example, a list of the first five U.S. presidents would look like this:

```
<ol>
   <li>George Washington</li>
   <li>John Adams</li>
   <li>Thomas Jefferson</li>
   <li>James Madison</li>
   <li>James Monroe<li>
</ol>
```

UNORDERED LIST

Unordered lists are used when the items in the list have no particular order, they are just a list of items that are of equal rank or importance. The basic building blocks are the element and the element(s) which represent the individual items in the list. For example, a list of items to get at the store would look like this:

```
<ul>
   <li>Milk</li>
   <li>Bread</li>
   <li>Eggs</li>
   <li>Bananas</li>
   <li>Shampoo</li>
</ul>
```

DESCRIPTION LIST

Description lists are used to provide a name – value pairing(s). Similar to what you would see in a dictionary, there is a name followed by one or more values or descriptions. The building blocks are the <dl></dl>, <dt></dt> and <dd></dd> elements. The <dl></dl> opens and closes the list, the <dt></dt> identified the name and the <dd></dd> begins and ends the value. A description list that identifies the primary digital colors used in monitors:

```
<dl>
   <dt>R</dt>
      <dd>Red – One of the three primary digital hues.
            Normally seen in the abbreviation RGB.</dd>
   <dt>G</dt>
      <dd>Green – One of the three primary digital hues.
            Normally seen in the abbreviation RGB.</dd>
   <dt>B</dt>
      <dd>Blue – One of the three primary digital hues.
            Normally seen in the abbreviation RGB.</dd>
</dl>
```

NESTING LISTS

Occasionally it is necessary to nest or place one list inside another. It is easy to do, just remember that when building an ordered or unordered list the only element that can be a child of the <ol> or <ul> is <li>. So, to nest a list, the nested list must be inside the <li> element. For example, a list of web pages that would be found inside of folders in a web site:

```
<ul>
   <li>css
      <ul>
         <li>print.css</li>
         <li>screen.css</li>
      </ul>
   </li>
   <li>about
      <ul>
         <li>about.php</li>
         <li>contact.php</li>
      </ul>
   </li>
</ul>
```

TABLE

Tables are handy when organizing data in tabular format (columns). The basic building blocks are <table></table>, <tr></tr>, <th></th> and <td></td>. Refer to the element list at the end of this document for an explanation of each element. Tables can be very complicated, but a simple table of the calendar quarters might look like this:

```
<table>
   <tr>
      <th>Quarter</th>
      <th>Months</th>
   </tr>
   <tr>
      <td>First</td>
      <td>January, February, March</td>
   </tr>
   <tr>
      <td>Second</td>
      <td>April, May, June</td>
   </tr>
   <tr>
      <td>Third</td>
      <td>July, August, September</td>
   </tr>
   <tr>
      <td>Fourth</td>
      <td>October, November, December</td>
   </tr>
</table>
```

Naming Rules, Images and Paths

NAMING RULES

While these naming rules are not formal rules in the typical sense, they are important to follow in order to save time, money and frustration. They represent conventions or "best practices" that designers and developers have developed over time.

When it comes to naming web assets (pages, images, movies, audio files, script files and more) there are four simple rules to follow:

1. Lower case – use no capital letters – ever!
2. No spaces – spaces should be left out completely or replaced with hyphens or underscores. (e.g. instead of about me.php, use about_me.php, about-me.php or even better – about.php).
3. No special characters – Slashes, apostrophes, exclamation points, question marks and the like all have other meanings to the operating system of the computer or the protocols used by computers. If they are included in the name of a file or asset, chances are problems will occur and things end up broke. Leave them out.
4. Short, meaningful names – Provide names that convey real meanings about the file they identify. But, at the same time, keep them short. Remember that someone will have to type them at some point. For example, if you are building a web page of pictures you took, the name "gallery" works great. If you are building a web page that contains your resume, then "resume" is perfect.

IMAGES

The old adage "A picture is worth a thousand words" is true. Images in a web site can add interest, illustrate your content, draw the client into the web page and add beauty and variety to the site. However, there are few best practices to follow when using images:

- Images should supplement or enhance the content. Don't use a picture of a blender when talking about exotic fish – the two just don't blend well (pun definitely intended).
- The image should be optimized. As small as possible to load quickly so the client doesn't have to wait and pay for unnecessary bandwidth consumption.
- Add "alt" attributes and meaningful values to images. Some clients can't see well or browse with images disabled. Alt values give a text description of the image. These should be short but descriptive. They are also a great place to add content keywords to help with search engine rankings for your site.

<img> ELEMENT

Many new web developers fail to understand that the images that appear in a web page don't reside in the web page. They are stored elsewhere and the browser downloads and inserts them into the web page on the fly. As such the code placed in the web page to display the image must have an accurate path to where the image is stored so that it can be found and retrieved correctly.

The <img> element indicates to the browser that an image is to be inserted at this location. At the very least it should have the element and two attributes – src and alt. The src attribute must contain the path to where the image is stored and the alt attribute contains the text description of the image. Say you are building a fan tribute site for the movie Napolean Dynamite. One of your pages is for Pedro, Napolean's friend, who is running for student body president. You decide a picture of Pedro wearing his "Pedro for President" shirt is a must. The code might look similar to:

<img src="/images/pedro.jpg" alt="Pedro wearing his Pedro for President shirt from Napolean Dynamite movie">

In addition, the height and width attributes can be added to the <img> element, to tell the browser the dimensions to set aside for the image. While this is allowed, in many cases this is a better done in CSS. If added using HTML5 the values for width and height are integers and the browser interprets the values as being pixels.

PATHS

As mentioned earlier, paths are used to describe where assets are stored within the structure of a web site. There are only two types of paths: absolute and relative.

Absolute Path

An absolute path describes the entire path to follow to find an asset. When describing a path within your web site, the beginning point in typically the root folder or where the entire web site is stored.

A forward slash "/" is the indicator that tells the browser that an absolute path from the root folder is being used. In the image path (above) you will note the forward slash inside the src attribute. This tells the browser that within the root folder is an images folder and within it is the actual image. Using this type of path also means the browser doesn't care or even need to know where the file is that the image is being placed into. This is the path to be used most often.

Relative Path

A relative page describes where an asset is in relation to the file that it will be used with. In short, you must know where both are located and be able to describe the path relative to both. Use this only when assets are in the same folder.

ELEMENT	DESCRIPTION	EXAMPLE	ATTRIBUTES
a	Hyperlink		Globals; href; target; rel; media; hreflang; type
abbr	Abbreviation	<abbr title=" "></abbr>	Globals
address	Contact information	<address></address>	Globals
article	Self-contained composition	<article></article>	Globals
aside	Sidebar for related content	<aside></aside>	Globals
audio	Audio player	<audio controls></audio>	Globals; src; crossorigin; preload; autoplay; mediagroup; loop; muted; controls
b	Keyword		Globals
base	Base URL or default for hyperlinks	<base href="">	Globals; href; target
blockquote	Section quoted from other content	<blockquote></blockquote>	Globals; cite
body	Document body	<body></body>	Globals; onafterprint; onbeforeprint; onbeforeunload; onblur; onerror; onfucus; onhashchange; onload; onmessage; onoffline; ononline; onpagehide; onpageshow; onpopstate; onredo; onresize; onscroll; onstorage; onundo; onunload
br	Line break	 	Globals
canvas	Scriptable bitmap drawing area	<canvas width="" height=""></canvas>	Globals; width; height
caption	Table caption	<caption></caption>	Globals
cite	Title of a work	<cite></cite>	Globals
col	Table column designator	<col span="">	Globals; span
colgroup	Group of table columns	<colgroup></colgroup>	Globals; span
dd	Content for corresponding dt element	<dd></dd>	Globals
details	An interactive widget for content that need not be visible always.	<details></details>	Globals; open
div	Generic container		Globals
dl	Definition list	<dl></dl>	Globals
dt	Legend for associated dd elements	<dt></dt>	Globals
em	Stress emphasis		Globals
fieldset	Group of form controls	<fieldset></fieldset>	Globals; disabled; form; name
figcaption	Caption for a figure element	<figcaption></figcaption>	Globals
figure	Illustrative content	<figure></figure>	Globals

ELEMENT	DESCRIPTION	EXAMPLE	ATTRIBUTES
footer	Conclusion area for a page or section	<footer></footer>	Globals
form	User submittable form	<form></form>	Globals; accept-charset; action; autocomplete; enctype; method; novalidate; target
h1, h2, h3, h4, h5, h6	Section headings (levels of importance)		Globals
head	Container for document metadata	<head></head>	Globals
header	Introductory area for a page or section	<header></header>	Globals
hr	Thematic break	<hr>	Globals
html	Root element	<html></html>	Globals; manifest
i	Alternate voice		Globals
img	Image	<img src="" alt="">	Globals; alt; src; crossorigin; usemap; ismap; width; height
input	Form control	<input type="" name="">	Globals; accept; alt; autocomplete; autofocus; checked; dirname; disabled; form; formaction; formenctype; formmethod; formnovalidate; formtarget; height; list; max; maxlength; min; multiple; name; pattern; placeholder; readonly; required; size; src; step; type; value; width
label	Caption for a form control	<label></label>	Globals; form; for
legend	Caption for a fieldset	<legend></legend>	Globals
li	List item for ol and ul lists		Globals; value
link	Link metadata	<link href="" rel="" type="">	Globals; href; rel; media; hreflang; type; sizes
main	Main content container in a page	<main></main>	Globals
meta	Text metadata	<meta name="" content="">	Globals; name; http-equiv; content; charset
nav	Navigational section container	<nav></nav>	Globals
noscript	Fallback content for script	<noscript></noscript>	Globals
ol	Ordered list		Globals; reversed; start
p	Paragraph		Globals

ELEMENT	DESCRIPTION	EXAMPLE	ATTRIBUTES
script	Embedded script	<script src=””></script>	Globals; src; async; defer; type; charset
section	Generic document section	<section></section>	Globals
span	Generic phrasing container		Globals
strong	Importance		Globals
style	Embedded styling information	<style></style>	Globals; media; type; scoped
table	Tabular data container	<table></table>	Globals; border
tbody	Group of data rows in a table	<tbody></tbody>	Globals
td	Table cell	<td></td>	Globals; colspan; rowspan; headers
textarea	Multiline form text field	<textarea></textarea>	Globals; autofocus; cols; disabled; form; maxlength; name; placehoder; readonly; required; rows; wrap
tfoot	Group of table footer rows	<tfoot></tfoot>	Globals
th	Table header cell	<th></th>	Globals; colspan; rowspan; headers; scope
thead	Group of table header rows	<thead></thead>	globals
title	Document title	<title></title>	Globals
tr	Table row	<tr></tr>	globals
ul	Unordered list		globals
video	Video player	<video controls></video>	globals; src; crossorigin; poster; preload; autoplay; mediagroup; loop; muted; controls; width; height